CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CE266: SOFTWARE ENGINEERING

Dec. 2023 – April 2024

UNIT 8

Software Maintenance and Configuration Management

DEPSTAR

Content

- Types of Software Maintenance, Re-engineering, Reverse Engineering, Forward Engineering
- The SCM process, Identification of Objects in the Software Configuration
- Version Control and Change Control

Software Maintenance

- Software is released to end-users, and
  - within days, bug reports filter back to the software engineering organization.
  - within weeks, one class of users indicates that the software must be changed so that it can accommodate the special needs of their environment.
  - within months, another corporate group who wanted nothing to do with the software when it was released, now recognizes that it may provide them with unexpected benefit. They'll need a few enhancements to make it work in their world.
- All of this work is *software maintenance*

Types of Software Maintenance

- **Corrective Maintenance**:
  - This includes modifications and updations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.
  - Corrective maintenance deals with the repair of faults or defects found in day-today system functions.
  - A defect can result due to errors in software design, logic and coding.
  - Design errors occur when changes made to the software are incorrect, incomplete, wrongly communicated, or the change request is misunderstood.

Contd…

- **Adaptive Maintenance:**
- This includes modifications and updations applied to keep the software product up-to date and tuned to the ever changing world of technology and business environment.
- Adaptive maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system.
- Adaptive maintenance consists of adapting software to changes in the environment such as the hardware or the operating system.
- The term environment in this context refers to the conditions and the influences which act (from outside) on the system.

Contd…

- **Perfective Maintenance:**

- This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.
- Perfective maintenance mainly deals with implementing new or changed user requirements.
- Perfective maintenance involves making functional enhancements to the system in addition to the activities to increase the system's performance even when the changes have not been suggested by faults.
- This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per the user's; changing needs.

Contd…

- **Preventive Maintenance:**
- This includes modifications and updations to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future.
- Preventive maintenance involves performing activities to prevent the occurrence of errors.
- It tends to reduce the software complexity thereby improving program understandability and increasing software maintainability. It comprises documentation updating, code optimization, and code restructuring.
- Documentation updating involves modifying the documents affected by the changes in order to correspond to the present state of the system.

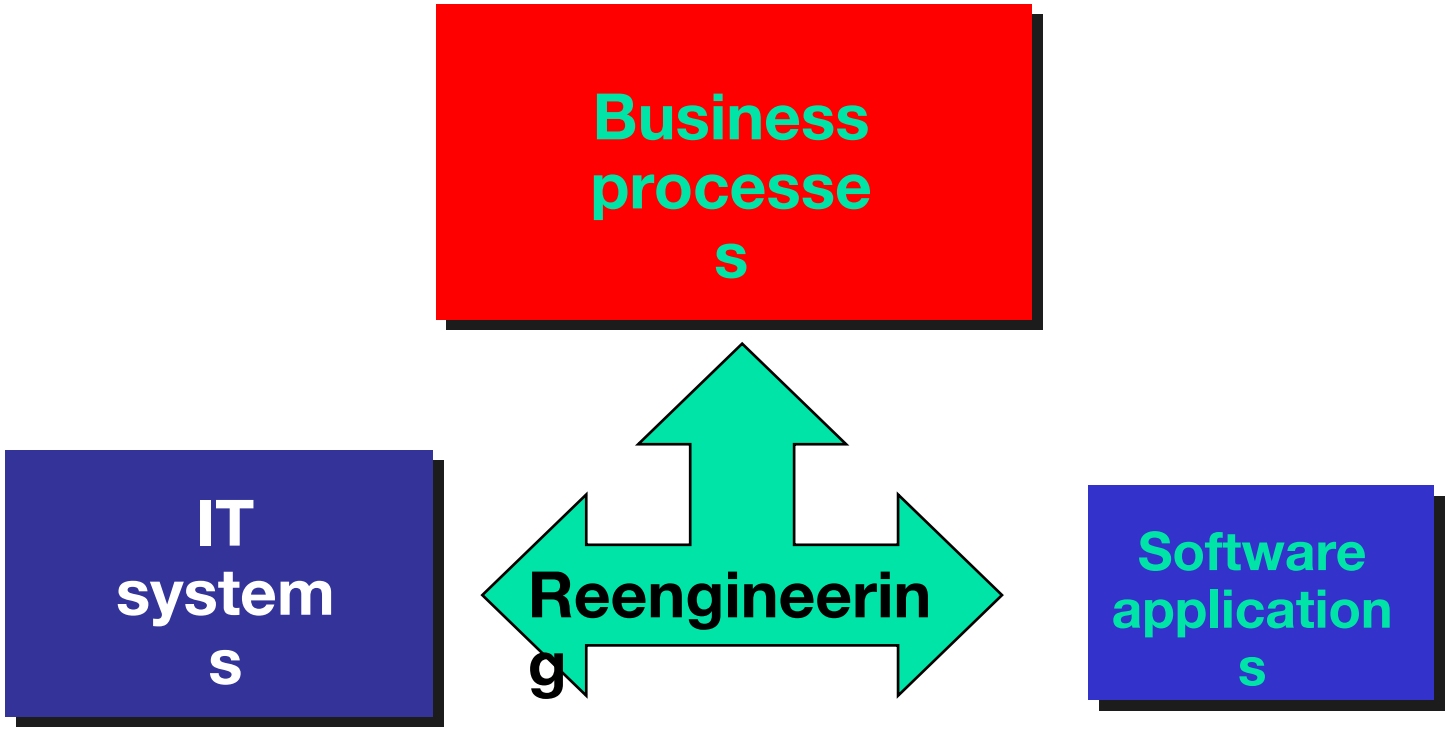
Maintainable Software

- Maintainable software exhibits effective modularity
- It makes use of design patterns that allow ease of understanding.
- It has been constructed using well-defined coding standards and conventions, leading to source code that is self-documenting and understandable.
- It has undergone a variety of quality assurance techniques that have uncovered potential maintenance problems before the software is released.
- It has been created by software engineers who recognize that they may not be around when changes must be made.
  - *Therefore, the design and implementation of the software must "assist" the person who is making the change*

Software Supportability

- "the capability of supporting a software system over its whole product life.
  - This implies satisfying any necessary needs or requirements, but also the provision of equipment, support infrastructure, additional software, facilities, manpower, or any other resource required to maintain the software operational and capable of satisfying its function." [SSO08]
- The software should contain facilities to assist support personnel when a defect is encountered in the operational environment (and make no mistake, defects *will* be encountered).
- Support personnel should have access to a database that contains records of all defects that have already been encountered—their characteristics, cause, and cure.

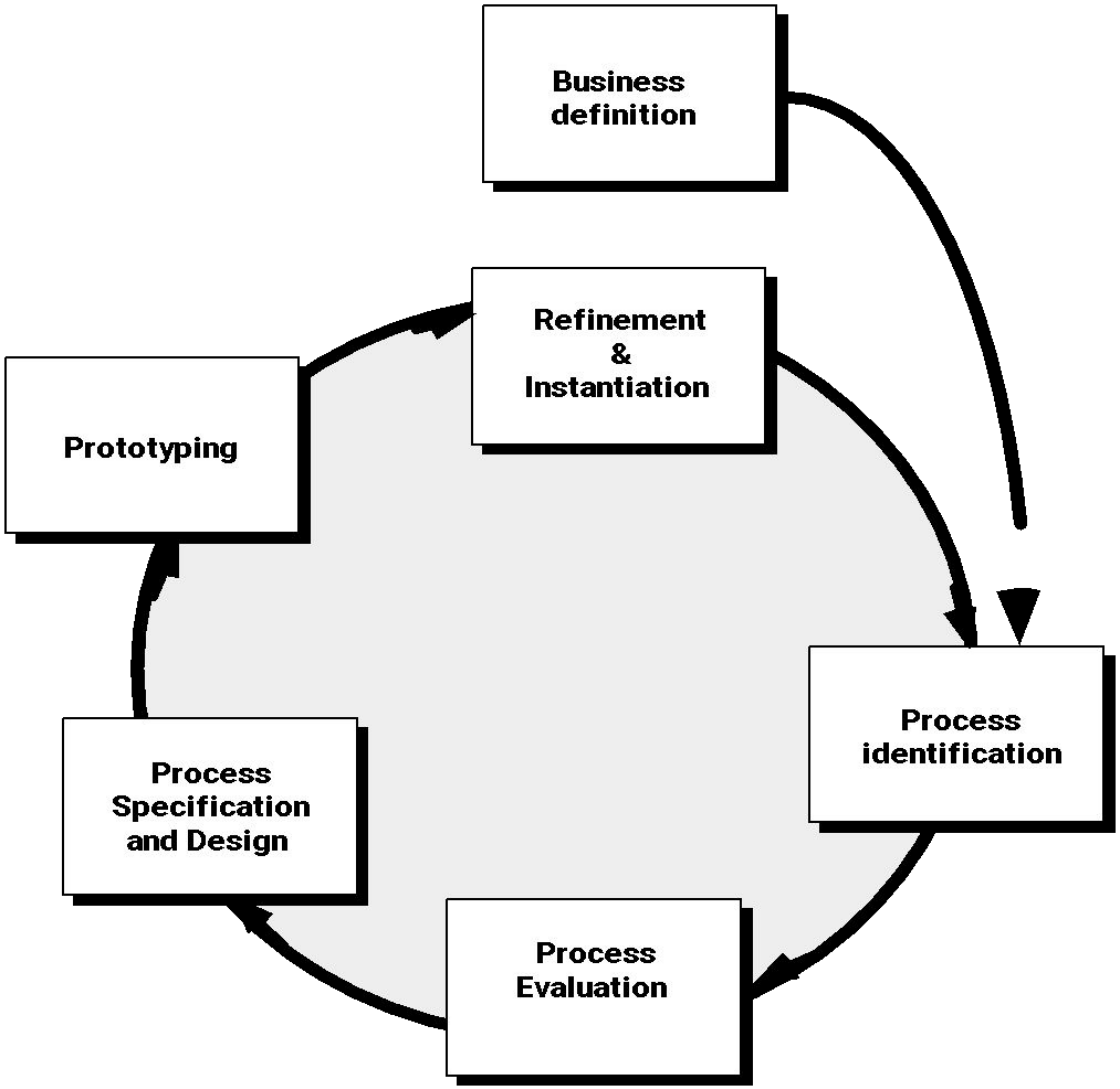
Reengineering



Business Process Reengineering

- **Business definition.** Business goals are identified within the context of four key drivers: cost reduction, time reduction, quality improvement, and personnel development and empowerment.
- **Process identification.** Processes that are critical to achieving the goals defined in the business definition are identified.
- **Process evaluation.** The existing process is thoroughly analyzed and measured.
- **Process specification and design.** Based on information obtained during the first three BPR activities, use-cases are prepared for each process that is to be redesigned.
- **Prototyping.** A redesigned business process must be prototyped before it is fully integrated into the business.
- **Refinement and instantiation.** Based on feedback from the prototype, the business process is refined and then instantiated within a business system.

Business Process Reengineering

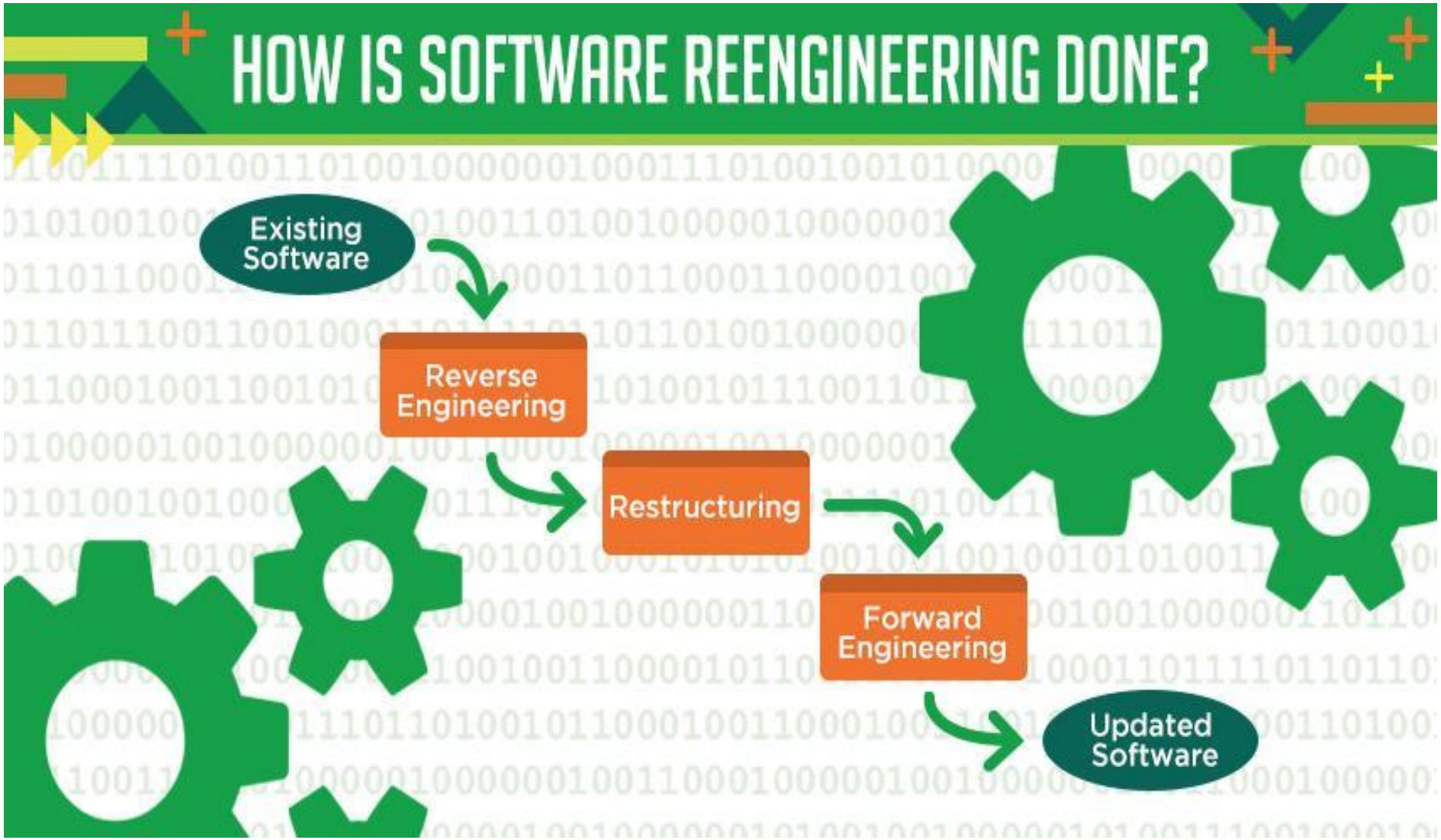


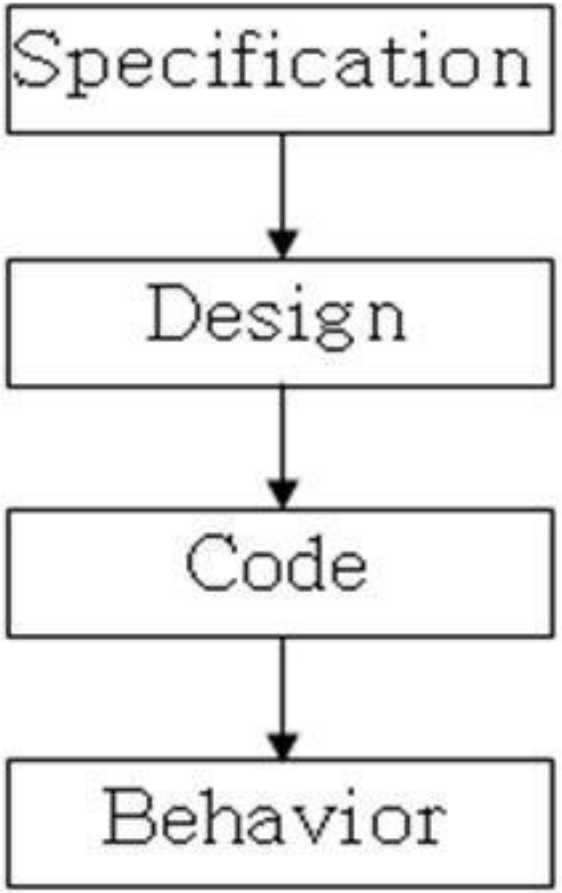
BPR Principles

- Organize around outcomes, not tasks.
- Have those who use the output of the process perform the process.
- Incorporate information processing work into the real work that produces the raw information.
- Treat geographically dispersed resources as though they were centralized.
- Link parallel activities instead of integrated their results. When different put the decision point where the work is performed, and build control into the process.
- Capture data once, at its source.

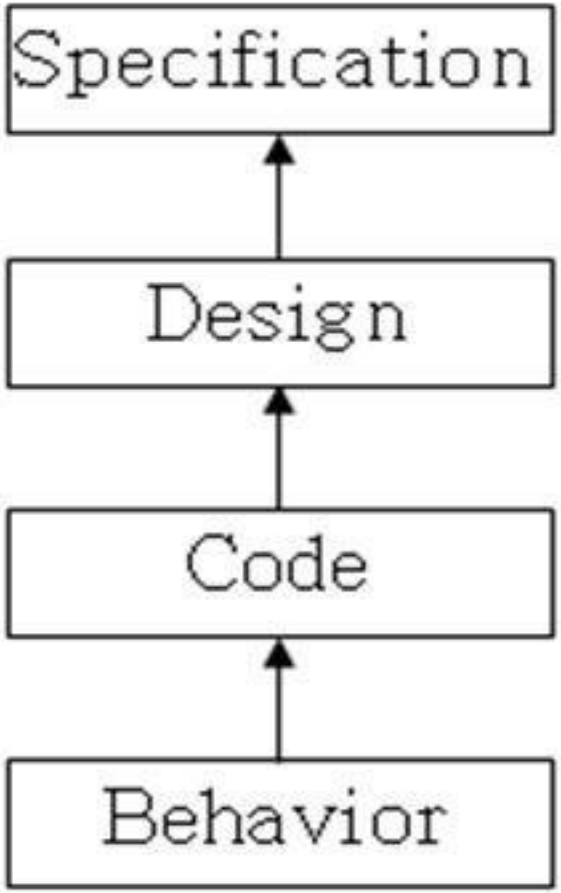
Software Reengineering

- **Software Reengineering** is the process of updating **software** without affecting its functionality.
- This process may be done by developing additional features on the **software** and adding functionalities that may or may not be required but considered to make the **software** experience better and more efficient.
- The need for software reengineering becomes an integral part of improving the quality of your products. This process adds more value to your business as it does not only better your services but also contribute added revenue.


HOW IS SOFTWARE REENGINEERING DONE?
Existing Software
Reverse Engineering
Restructuring
Forward Engineering
Updated Software



Forward Engineering

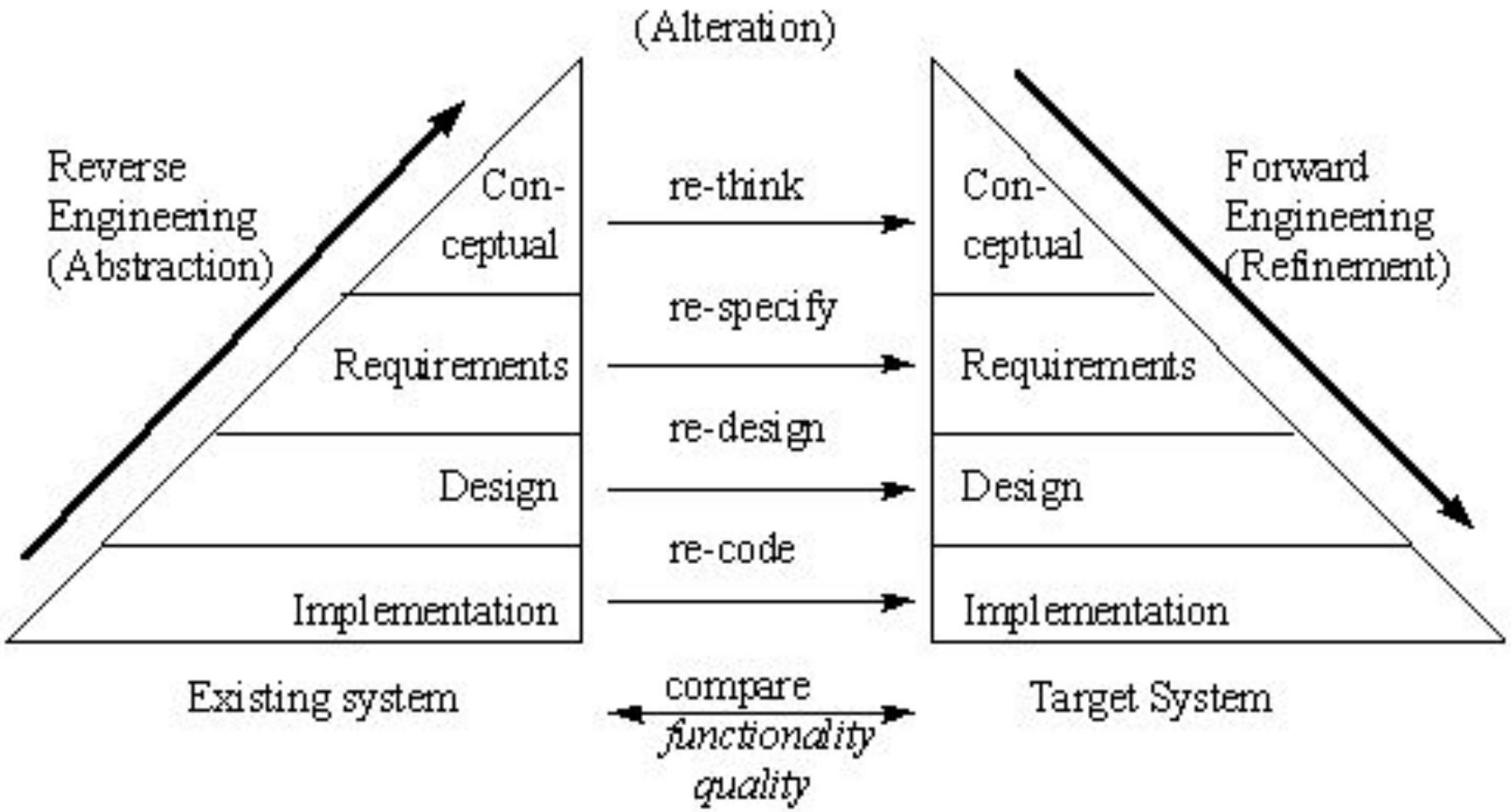


Reverse Engineering

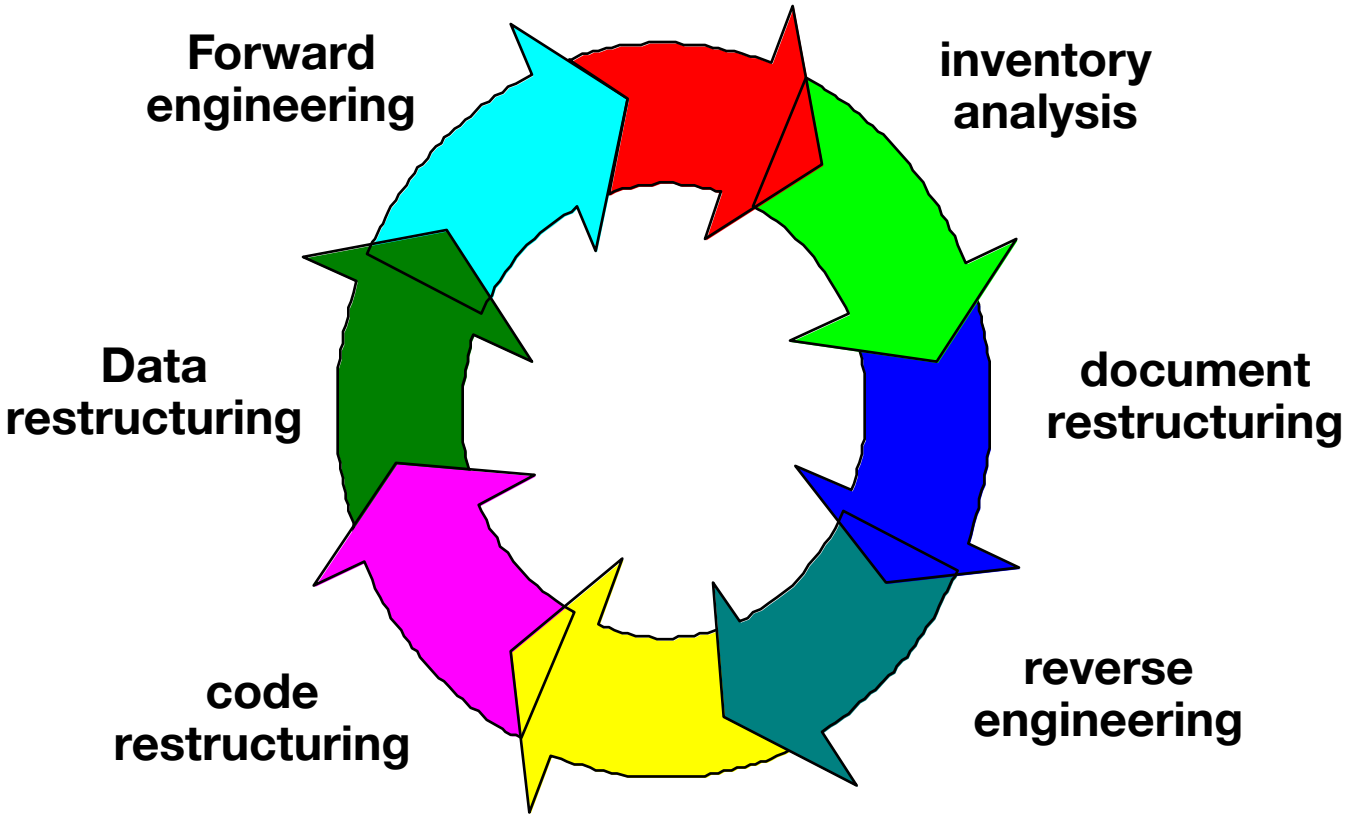
Forward Engineering

Vs

Reverse Engineering


(Alteration)
Reverse
Engineering
(Abstraction)
Con-
ceptual
Requirements
Design
Implementation
re-think
re-specify
re-design
re-code
Con-
ceptual
Requirements
Design
Implementation
Forward
Engineering
(Refinement)
Existing system
compare
functionality
quality
Target System

Software Reengineering



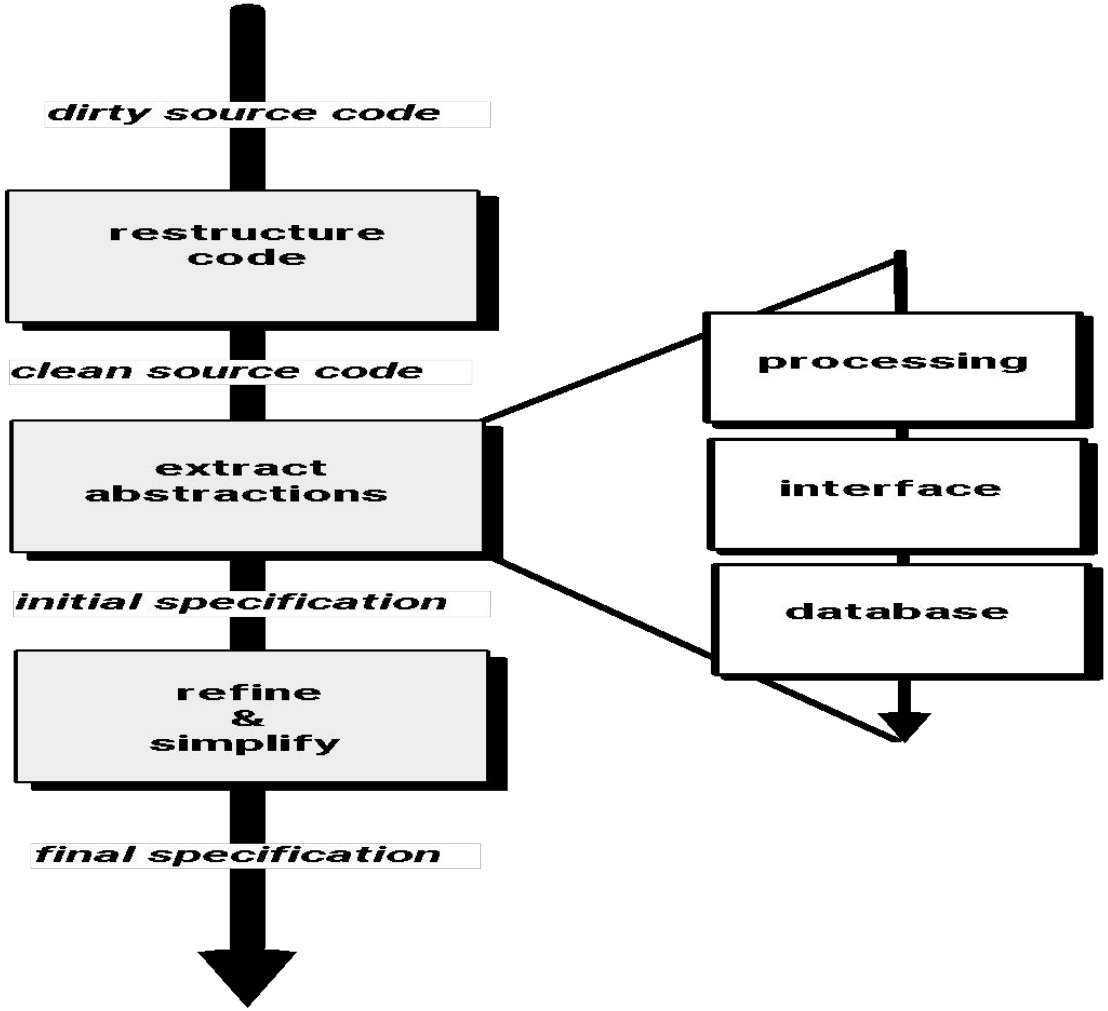
Inventory Analysis

- build a table that contains all applications
- establish a list of criteria, e.g.,
  - name of the application
  - year it was originally created
  - number of substantive changes made to it
  - total effort applied to make these changes
  - date of last substantive change
  - effort applied to make the last change
  - system(s) in which it resides
  - applications to which it interfaces, ...
- analyze and prioritize to select candidates for reengineering

Document Restructuring

- Weak documentation is the trademark of many legacy systems.
- But what do we do about it? What are our options?
- Options …
  - *Creating documentation is far too time consuming.* If the system works, we'll live with what we have. In some cases, this is the correct approach.
  - *Documentation must be updated, but we have limited resources.* We'll use a "document when touched" approach. It may not be necessary to fully redocument an application.
  - *The system is business critical and must be fully redocumented.* Even in this case, an intelligent approach is to pare documentation to an essential minimum.

Reverse Engineering



Code Restructuring

- Source code is analyzed using a restructuring tool.
- Poorly design code segments are redesigned
- Violations of structured programming constructs are noted and code is then restructured (this can be done automatically)
- The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced
- Internal code documentation is updated.

Data Restructuring

- Unlike code restructuring, which occurs at a relatively low level of abstraction, data structuring is a full-scale reengineering activity
- In most cases, data restructuring begins with a reverse engineering activity.
  - Current data architecture is dissected and necessary data models are defined (Chapter 9).
  - Data objects and attributes are identified, and existing data structures are reviewed for quality.
  - When data structure is weak (e.g., flat files are currently implemented, when a relational approach would greatly simplify processing), the data are reengineered.
- Because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.

Forward Engineering

1. The cost to maintain one line of source code may be 20 to 40 times the cost of initial development of that line.

2. Redesign of the software architecture (program and/or data structure), using modern design concepts, can greatly facilitate future maintenance.

3. Because a prototype of the software already exists, development productivity should be much higher than average.

4. The user now has experience with the software. Therefore, new requirements and the direction of change can be ascertained with greater ease.

5. CASE tools for reengineering will automate some parts of the job.

6. A complete software configuration (documents, programs and data) will exist upon completion of preventive maintenance.

Economics of Reengineering

- A cost/benefit analysis model for reengineering has been proposed by Sneed [Sne95]. Nine parameters are defined:
  - P_1 = current annual maintenance cost for an application.
  - P_2 = current annual operation cost for an application.
  - P_3 = current annual business value of an application.
  - P_4 = predicted annual maintenance cost after reengineering.
  - P_5 = predicted annual operations cost after reengineering.
  - P_6 = predicted annual business value after reengineering.
  - P_7 = estimated reengineering costs.
  - P_8 = estimated reengineering calendar time.
  - P_9 = reengineering risk factor (P_9 = 1.0 is nominal).
  - L = expected life of the system.

Contd…

- The cost associated with continuing maintenance of a candidate application (i.e., reengineering is not performed) can be defined as

$$C_{maint} = [P_3 - (P_1 + P_2)] \times L$$

- The costs associated with reengineering are defined using the following relationship:

$$C_{reeng} = [P_6 - (P_4 + P_5) \times (L - P_8) - (P_7 \times P_9)]$$

- Using the costs presented in equations above, the overall benefit of reengineering can be computed as

$$\text{cost benefit} = C_{reeng} - C_{maint}$$

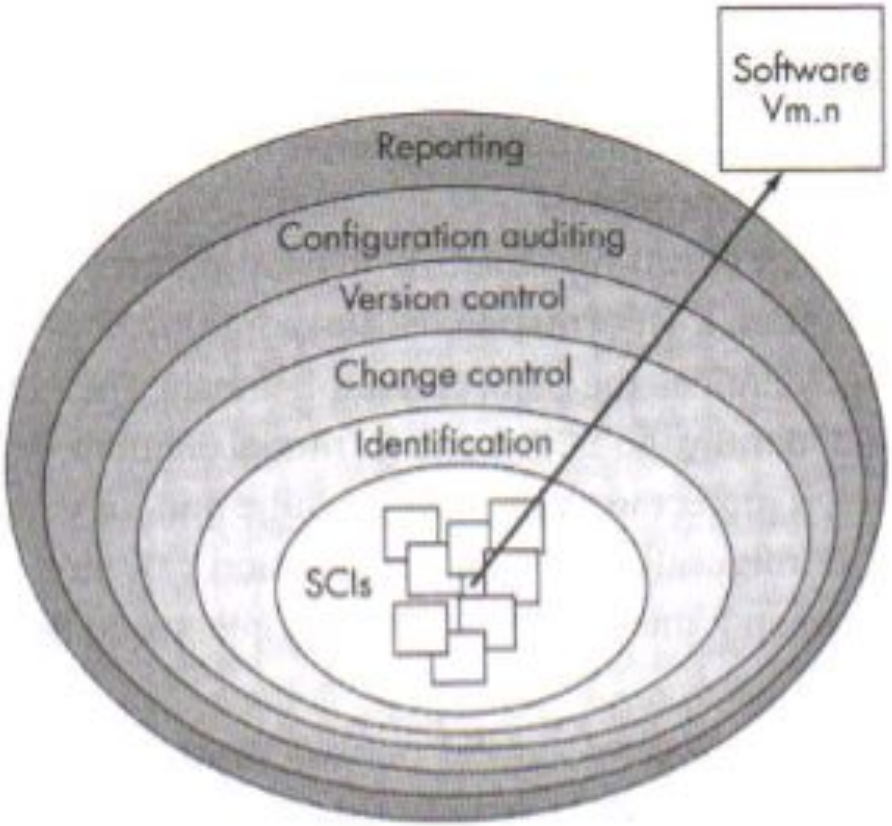
The SCM (Software Configuration Management) Process

- The software configuration management process defines a series of tasks that have four primary objectives:

1. To identify all items that collectively define the software configuration.
2. To manage changes to one or more of these items.
3. To facilitate the construction of different versions of an application.
4. To ensure that software quality is maintained as the configuration evolves over time.
5. Referring to the figure, SCM tasks can viewed as concentric layers.

- SCIs (Software Configuration Item) flow outward through these layers throughout their useful life, ultimately becoming part of the software configuration of one or more versions of an application or system.
- As an SCI moves through a layer, the actions implied by each SCM task may or may not be applicable. For example, when a new SCI is created, it must be identified.
- However, if no changes are requested for the SCI, the change control layer does not apply. The SCI is assigned to a specific version of the software (version control mechanisms come into play).
- A record of the SCI (its name, creation date, version designation, etc.) is maintained for configuration auditing purposes and reported to those with a need to know.

Layers of SCM Process



Identification of Objects in the Software Configuration

- To control and manage software configuration items, each should be separately named and then organized using an object-oriented approach.
- Two types of objects can be identified: basic objects and aggregate objects.
- A basic object is a unit of information that you create during analysis, design, code, or test.
- For example, a basic object might be a section of a requirements specification, part of a design model, source code for a component, or a suite of test cases that are used to exercise the code.
- An aggregate object is a collection of basic objects and other aggregate objects.
- For example, a Design Specification is an aggregate object. Conceptually, it can be viewed as a named (identified) list of pointers that specify aggregate objects such as Architectural Model and Data Model, and basic objects such as Component and UML Class Diagram.
- Each object has a set of distinct features that identify it uniquely: a name, a description, a list of resources, and a "realization."

Contd…

- The object name is a character string that identifies the object unambiguously.
- The object description is a list of data items that identify the SCI type (e.g., model element, program, data) represented by the object, a project identifier, and change and/or version information.
- Resources are "entities that are provided, processed, referenced or otherwise required by the object".

Version Control

- Version control combines procedures and tools to manage different versions of configuration objects that are created during the software process.
- A version control system implements or is directly integrated with four major capabilities:

1. A project database (repository) that stores all relevant configuration objects.
2. A version management capability that stores all versions of a configuration object.
3. A make facility that enables you to collect all relevant configuration objects and construct a specific version of the software.

- In addition, version control and change control systems often implement an issues tracking capability that enables the team to record and track the status of all outstanding issues associated with each configuration object.
- A number of version control systems establish a change set—a collection of all changes (to some baseline configuration) that are required to create a specific version of the software.
- A change set "captures all changes to all files in the configuration along with the reason for changes and details of who made the changes and when."
- A number of named change sets can be identified for an application or system.
- This enables you to construct a version of the software by specifying the change sets (by name) that must be applied to the baseline configuration.

- To accomplish this, a system modeling approach is applied. The system model contains:

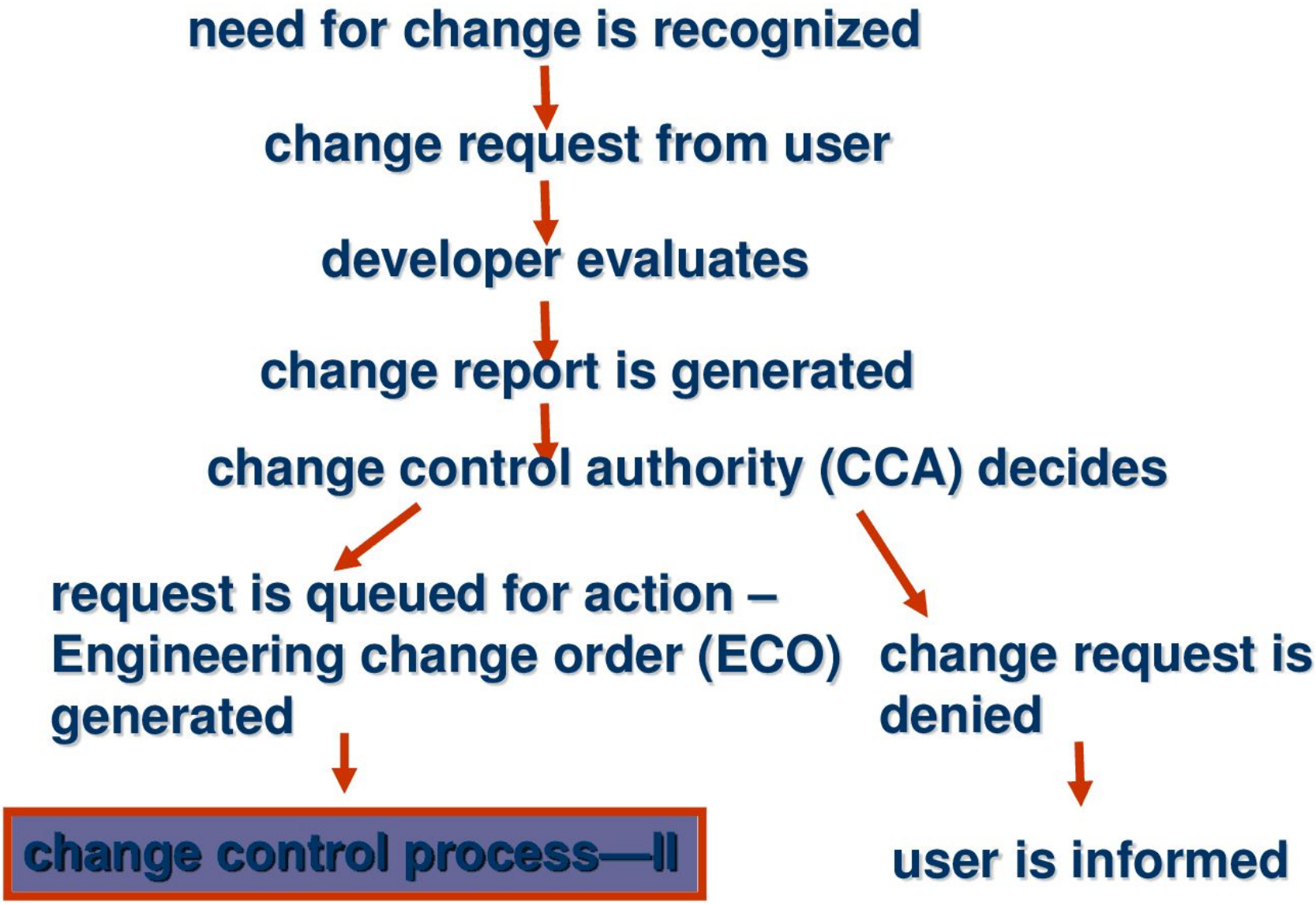
1. A template that includes a component hierarchy and a "build order" for the components that describes how the system must be constructed.
2. Construction rules.
3. Verification rules.

- The primary difference in approaches is the sophistication of the attributes that are used to construct specific versions and variants of a system and the mechanics of the process for construction.

Change Control

- For a large software project, uncontrolled change rapidly leads to chaos.
- For such projects, change control combines human procedures and automated tools to provide a mechanism for the control of change.
- A change request is submitted and evaluated to assess technical merit, potential side effects, overall impact on other configuration objects and system functions, and the projected cost of the change.
- The results of the evaluation are presented as a change report, which is used by a change control authority
- (CCA)—a person or group that makes a final decision on the status and priority of the change.
- An engineering change order (ECO) is generated for each approved change.
- The ECO describes the change to be made, the constraints that must be respected, and the criteria for review and audit.

Change Control Process—I




Thank you!